

From Uncoded Prefetching to Coded Prefetching in Coded Caching

Chao Tian and Kai Zhang

April 27, 2017

Abstract

In the coded caching framework proposed by Maddah Ali and Niesen, there are two classes of coding schemes known in the literature, namely uncoded prefetching schemes and coded prefetching schemes. In this work, we provide a connection between the uncoded prefetching scheme proposed by Maddah Ali and Niesen (and its improved version by Yu *et al.*) and the coded prefetching scheme proposed by Tian and Chen, when the number of files is no larger than that of users. We make a critical observation that a coding component in the Tian-Chen scheme can be replaced by a binary code, which enables us to view the two schemes as the extremes of a more general scheme. The intermediate operating points of this general scheme can in fact provide new tradeoff points previously not known in the literature, however, explicit characterizing the performance of this general scheme appears rather difficult.

1 Introduction

Caching can be used to relieve contention on communication resources by prefetching data to a local or fast memory space, and thus avoiding data retrieval from the remote or slower data source during peak traffic time. Traditionally, caching has mainly been considered in the single user setting, *e.g.*, on-CPU caches vs. RAM in computers, where the hit-ratio is the key measure of performance. As networked systems become more prevalent, caching involving multiple users has attracted increasingly more research attention.

In their award-winning paper [1], Maddah-Ali and Niesen provided a formal information theoretic formulation for the caching problem in multiuser settings; see Fig. 1. In this formulation, there are N files, each of F bits, and K users. Each user has a local cache memory of capacity MF (thus a normalized capacity of M). In the placement (or sometimes referred to as the prefetching) phase, the users can fill their caches with contents from the central server without the knowledge of the precise requests. In the delivery phase, each user reveals the request for a single file from the central server, and the central server must multicast certain common (and possibly coded) information to all the users in order to accommodate these requests. Since in the placement phase, the requests at the later phase are unknown a-priori, the cached contents must be strategically prepared at all the users. The goal is to minimize the amount of multicast information which has rate RF (or equivalently the normalized rate of R), under the normalized constraint on cache memory M . It was shown in [1] that coding can be rather beneficial, while uncoded solutions suffer a significant loss. Subsequent works extended it also to decentralized caching placements [2], caching with nonuniform demands [3], online caching placements [4], and hierarchical coded caching [5].

There were quite a few recent efforts [6–13] aiming to finding better codes with improved memory-rate tradeoff. In particular, Yu *et al.* [11] proposed a strategy that is optimal when the prefetching is uncoded, which in fact directly improved on the scheme in [1]. The key insight in [11] appears to be that the original delivery strategy in [1] may have redundancy in the transmissions,

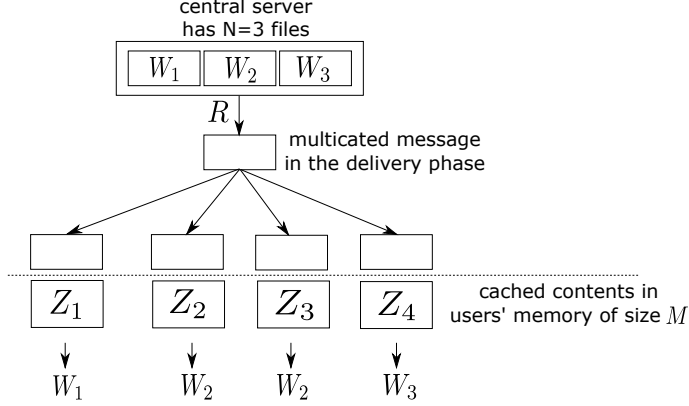


Figure 1: An example caching system instance, where there are $N = 3$ files, denoted as (W_1, W_2, W_3) , and $K = 4$ users, whose cached contents are (Z_1, Z_2, Z_3, Z_4) , respectively. In this instance the users request files $(1, 2, 2, 3)$, respectively, the multicast information together with the cached contents at each user, can be used to recovered the requested files.

which can be systematically removed to directly improve the the delivery rate. In another recent work, Tian and Chen [10] proposed a coded prefetching and the corresponding delivery strategy when $N \leq K$, which relies on a combination of rank metric code and maximum distance separable (MDS) codes in a non-binary finite field. In the regime when the memory size M is relatively small, the scheme in [10] can achieve a better performance than that in [11].

In this work, we consider this centralized caching problem, and show that the scheme in [10] can be slightly modified, where the MDS codes used in the delivery phase can be replaced by a code using only binary additions (XOR). Though the alternative construction itself does not provide further improvement on the memory-rate tradeoff, it allows us to make a conceptual connection between the scheme in [10] and that in [11]. It further enables us to view these two schemes as the two extremes of a more general of scheme. The intermediate operating points of this more general scheme can indeed provide new tradeoff points previously not known in the literature. We provide details for one particular example of this improvement for the $(N, K) = (3, 4)$ case, however the precise tradeoffs achieved by the general scheme turn out to be difficult to characterize analytically.

The rest of the paper is organized as follows. Section 2 provides some necessary reviews on existing results and rank metric codes. A critical observation behind the alternative code is given in Section 3, followed by the code constructions and the proof in 4. Section 5 gives one special case where an intermediate code provides a new tradeoff point, and the difficulty of characterizing the general case performance is also discussed. Section 6 concludes the paper with a few remarks.

2 Known Results and Preliminaries

We first briefly review the existing relevant results on the coded caching problem, and provide necessary results on rank metric codes, which serve an instrumental role in the new code construction.

2.1 Schemes Using Uncoded Prefetching

The scheme in [1] can achieve the following tradeoff pair

$$(M, R) = \left(\frac{tN}{K}, \frac{K-t}{1+t} \right), \quad t = 0, 1, \dots, K, \quad (1)$$

and since another trivial point is clearly $(M, R) = (0, N)$, the lower convex hull of them provides an upper bound to the optimal tradeoff, as stated in [1]. More recently, Yu *et al.* [11] gave an optimal scheme when the prefetching is restricted to be uncoded, and the tradeoff achieved by this improved version of the scheme in [1] is

$$(M, R) = \left(\frac{tN}{K}, \frac{\binom{K}{t+1} - \binom{K - \min\{K, N\}}{t+1}}{\binom{K}{t}} \right), \quad t = 0, 1, \dots, K, \quad (2)$$

which strictly improves upon (1) when $K - N \geq t + 1$. Both schemes in [1] and [11] use the same uncoded placement strategy, but slightly different delivery strategies, which can be roughly described as follows.

Choose a fixed t in the scheme, and partition each file into $\binom{K}{t}$ segments of equal size; each segment is thus uniquely associated with a cardinality- t subset \mathcal{A} of the user set $\{1, 2, \dots, K\}$, and this segment is placed in the caches of users in \mathcal{A} . During delivery, consider each $(t + 1)$ subset \mathcal{B} of users: each user is requesting a segment that is in all the other users' cache within this group, and the server thus sends the XOR of all such segments of this group. Each user can recover the desired segment since all other segments involved are known to this user. As mentioned earlier, these transmissions may in fact have redundancy (*i.e.*, some can be uniquely determined from the others, or equivalently, they are linearly dependent) for certain parameters, and eliminating such redundancy results in the scheme in [11].

Let us consider an example with $N = 3$ files, denoted as A, B, C , respectively, and $K = 4$ users. Set the auxiliary variable $t = 2$, then each file is partitioned into $\binom{4}{2} = 6$ segments, for example, for file A , into $A_{1,2}, A_{1,3}, A_{1,4}, A_{2,3}, A_{2,4}, A_{3,4}$, and in this set segment $A_{1,2}$ is given to users 1 and 2, etc.. Suppose now the request is (A, A, B, C) , *i.e.*, the first two users request file A , the third user requests file B , and the fourth user requests file C . Consider the subset of users $\mathcal{B} = \{1, 2, 3\}$, for which we can transmit $A_{2,3} + A_{1,3} + B_{1,2}$; clearly the users in \mathcal{B} can recover $A_{2,3}, A_{1,3}, B_{1,2}$, respectively. For other subsets of users, the transmissions are formed similarly, and thus the complete transmissions are

$$\begin{aligned} &A_{2,3} + A_{1,3} + B_{1,2}, A_{2,4} + A_{1,4} + C_{1,2}, \\ &A_{3,4} + B_{1,4} + C_{1,3}, A_{3,4} + B_{2,4} + C_{2,3}, \end{aligned} \quad (3)$$

where the addition is in the binary field. The transmissions of (3) do not have redundancy in this particular case.

The schemes in both [8] and [9] use uncoded prefetching, and since the scheme in [11] is optimal for this class of codes, these two schemes in [8] and [9] do not provide any additional improvement over (2).

2.2 Schemes Using Coded Prefetching

Even in the pioneering work [1], it was observed that uncoded prefetching scheme is not sufficient. One coded placement scheme was provided for the case $(N, K) = (2, 2)$, and the tradeoff it achieves is indeed optimal. In [6], Chen *et al.* extended this scheme to the general case $N \leq K$, and showed that $\left(\frac{1}{K}, \frac{N(K-1)}{K} \right)$ is achievable and optimal.

More recently, Tian and Chen [10] proposed a more general scheme with coded prefetching for $N \leq K$. It was shown that the scheme can achieve the tradeoff pairs

$$(M, R) = \left(\frac{t[(N-1)t + K - N]}{K(K-1)}, \frac{N(K-t)}{K} \right), \quad t = 0, 1, \dots, K. \quad (4)$$

With $t = 1$, it reduces to the tradeoff point given in [6].

The scheme in [10] is somewhat involved, but the digest is as follows. Each file is again partitioned into $\binom{K}{t}$ segments of equal size, and given to the relevant users as in [11]; however, instead of directly storing them, each user caches certain linear combinations of these corresponding segments, mixed across all the files. During delivery, each symbol being transmitted is a linear combination of the segments from a single file, that serves two roles: firstly, the segments forming a single linear combination being transmitted are all present at certain user's cache that is not requesting this file, thus this user can use it to help resolve the cached symbols when sufficient such transmissions are received; secondly, these segments are not present at some users which are requesting that file, thus can also help them to recover the missing segments.

Let us consider again the example $(N, K) = (3, 4)$ and $t = 2$. In this case, the linear combinations of the segments

$$A_{1,2}, A_{1,3}, A_{1,4}, B_{1,2}, B_{1,3}, B_{1,4}, C_{1,2}, C_{1,3}, C_{1,4} \quad (5)$$

are placed at user 1's cache, where each segment is viewed as a symbol in a large finite field. According to the scheme in [10], there should be a total of 5 linear combinations cached; the coefficients of these linear combinations are not essential in this construction, and random assignments will be valid with high probability in a sufficiently large finite field. Now consider again the requests (A, A, B, C) . In this case, the server will send the following 8 symbols

$$\begin{aligned} &A_{3,4}, B_{1,2}, B_{1,4}, B_{2,4}, C_{1,2}, C_{1,3}, C_{2,3}, \\ &A_{1,3} + A_{2,3}, A_{1,4} + A_{2,4}, \end{aligned} \quad (6)$$

where the addition is in the same finite field of the information symbol which is usually not in the binary field; moreover, in general the linear combinations in the transmissions may need to use coefficients that are neither 0 nor 1 in the scheme of [10]. Now user 1 collects from (6) the symbols $B_{1,2}, B_{1,4}, C_{1,2}, C_{1,3}$, which, together with 5 cached linear combinations, leads to a total of 9 linear combinations of the basis in (5). Since the linear combinations are made linearly independent, all of the symbols can be resolved. User 1 then collects $A_{1,3} + A_{2,3}, A_{1,4} + A_{2,4}$ from which $A_{2,3}$ and $A_{2,4}$ can be recovered by eliminating $A_{1,3}$ and $A_{1,4}$. It can be verified in a similar manner that all other users can also recover the requested files, and for any other type of requests of the files, transmissions of 8 symbols will always suffice.

Amiri and Gunduz [12] showed that the following tradeoff point is achievable when $N \leq K$

$$(M, R)^* = \left(\frac{N-1}{K}, \frac{N(2K-N)}{2K} \right). \quad (7)$$

However, it can be verified that $(M, R)^*$ is precisely on the time-sharing line between (2) and (4) with $t = 1$. More recently, Gomez-Vilardebo [13] showed that the following tradeoff pairs are achievable:

$$(M, R) = \left(\frac{N}{Kg}, N - \frac{N(N+1)}{K(g+1)} \right), \quad g = 1, \dots, N. \quad (8)$$

The lower convex hull of (2), (4) and (8) provides the best known inner bound of the optimal (M, R) tradeoff in the literature.

2.3 Linearized Polynomial and Rank Metric Codes

Similar as in [10], rank metric codes based on linearized polynomials (see [14]) can be used to facilitate our code constructions. The following lemma is particularly relevant; see, *e.g.*, [15].

Lemma 1. *A linearized polynomial in finite field \mathbb{F}_{q^m}*

$$f(x) = \sum_{i=1}^P v_i x^{q^{i-1}}, \quad v_i \in \mathbb{F}_{q^m} \quad (9)$$

can be uniquely identified from evaluations at any P points $x = \theta_i \in \mathbb{F}_{q^m}$, $i = 1, 2, \dots, P$, that are linearly independent over \mathbb{F}_q .

Another relevant property of linearized polynomials is that they satisfy the following condition

$$f(ax + by) = af(x) + bf(y), \quad a, b \in \mathbb{F}_q, \quad x, y \in \mathbb{F}_{q^m}, \quad (10)$$

which is the reason that they are called “linearized”. This property implies the following lemma, the proof of which can be found in [10].

Lemma 2. *Let $f(x)$ be a linearized polynomial in \mathbb{F}_{q^m} as given in (9), and let $\theta_i \in \mathbb{F}_{q^m}$, $i = 1, 2, \dots, P_o$, be linearly independent over \mathbb{F}_q . Let G be a $P_o \times P$ full rank (rank P) matrix with entries in \mathbb{F}_q , then $f(x)$ can be uniquely identified from*

$$[f(\theta_1), f(\theta_2), \dots, f(\theta_{P_o})] \cdot G. \quad (11)$$

With a fixed set of $\theta_i \in \mathbb{F}_{q^m}$, $i = 1, 2, \dots, P_o$, which are linearly independent, we can view (v_1, \dots, v_P) as information symbols to be encoded, and the evaluations $[f(\theta_1), f(\theta_2), \dots, f(\theta_{P_o})]$ as the coded symbols. This is a (P_o, P) MDS code in terms of rank metric [14], where $P_o \geq P$. More importantly, the above lemma says any full rank (rank P) \mathbb{F}_q linear combinations of the coded symbols are sufficient to decode all the information symbols. This linear-transformation-invariant property had been utilized previously in other coding problems such as network coding with errors and erasures [16], locally repairable codes with regeneration [17], and layered regenerating codes [18].

The codes thus obtained are not systematic, but they can be converted to systematic codes by viewing the information symbols (w_1, w_2, \dots, w_P) as the first P evaluations $[f(\theta_1), f(\theta_2), \dots, f(\theta_P)]$, which can be used to find the coefficients of the linearized polynomial (v_1, v_2, \dots, v_P) , and then the additional parity symbols can be generated by evaluating this linearized polynomial at the remaining points $(\theta_{P+1}, \dots, \theta_{P_o})$. Systematic rank-metric codes are instrumental in our construction.

3 A Connection Between Coded and Uncoded Prefetching

The two schemes in [10] and [11] may seem very different at the first sight: one uses coded prefetching and the other uncoded, and one is non-binary code while the other is binary. Nevertheless, a closer look reveals some curious connections between the two codes. For example, the tradeoff points in (4) lead to the rate values $R = \frac{N(K-t)}{K}$ for $t = 0, 1, \dots, K$, which are exactly the same set of M values given in (2). This connection may or may not be a simple coincidence, however we next describe a much less obvious observation which leads to the main result of this paper.

Consider again the example case for $(N, K) = (3, 4)$ and $t = 2$. Let us decompose the transmissions in (3) by separating different files in the same linear combination. For example, the linear combination $A_{2,3} + A_{1,3} + B_{1,2}$ is decomposed into a pair of transmissions $(A_{2,3} + A_{1,3}, B_{1,2})$. It can be verified that decomposing all the linear combinations in (3) in fact produces exactly the same set of linear combinations in (6), after removing the repeated transmissions. We note that the addition in (6) is in a non-binary finite field, while the addition in (3) is in the binary field. However, as we show next, if we choose a binary extension field \mathbb{F}_{2^m} for the code in [10], the delivery can be

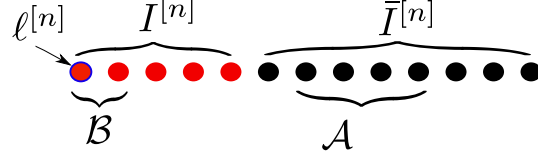


Figure 2: The relation of various sets.

accomplished using only additions of the information symbols in this binary extension field, *i.e.*, the coefficients of the linear combinations are either 0 or 1. Clearly, such additions are equivalent to additions in the base binary field, when we represent the information and code symbols in their binary vector form.

4 An Alternative Scheme with Coded Prefetching

Extending the observation given in the previous section, we now provide an alternative version of the Tian-Chen scheme where the delivery uses only binary additions. In the rest of the paper, the set of integers $\{1, 2, \dots, n\}$ is written as I_n , and the cardinality of a set \mathcal{A} is written as $|\mathcal{A}|$. Denote the N files in the system as W_1, W_2, \dots, W_N . Fix an integer parameter $t \in \{1, 2, \dots, K\}$ in the code, then in our code each file is partitioned into $\binom{K}{t}$ segments of equal size, and each file segment will be placed at t users, possibly as a component of some linear combinations. Each segment $W_{n,\mathcal{S}}$, where $n \in I_N$ and $\mathcal{S} \subseteq I_K$ with $|\mathcal{S}| = t$, is assumed to be a symbol in \mathbb{F}_{2^m} for some sufficiently large m to be specified later. Further define the following two parameters which will be useful in the construction

$$P \triangleq \binom{K-1}{t-1}N, \quad P_o \triangleq 2 \binom{K-1}{t-1}N - \binom{K-2}{t-1}(N-1).$$

4.1 Placement

The placement is identical to that in [10]. More precisely, user k collects the file segments:

$$\{W_{n,\mathcal{S}}, \text{ for all } n \in I_N, \text{ and all } \mathcal{S} \text{ such that } k \in \mathcal{S} \text{ and } |\mathcal{S}| = t\} \quad (12)$$

and then caches $P_o - P$ linear combinations (in the finite field \mathbb{F}_{2^m}) of these P file segments; we will specify the coefficients of the linear combinations later.

4.2 Delivery: All Files Requested

For a given demand vector (d_1, d_2, \dots, d_K) where the k -th user requests W_{d_k} , we define

$$I^{[n]} \triangleq \{k \in I_K : \text{user } k \text{ requests file } W_n\}, \quad n = 1, 2, \dots, N, \quad (13)$$

and denote the complementary set $\bar{I}^{[n]} \triangleq I_K \setminus I^{[n]}$. Let us for now focus on the case $m_n = |I^{[n]}| \geq 1$, $n = 1, 2, \dots, N$, *i.e.*, when all files are being requested; as showed in [10], all other cases can be addressed by slight variations from the solution in this case, and we shall revisit this issue later on. Fix also an arbitrary element (for example, the minimum element of $I^{[n]}$) in $I^{[n]}$, denoted as $\ell^{[n]}$, as the leader of $I^{[n]}$.

Suppose the demand is given by a fixed vector (d_1, \dots, d_K) , *i.e.*, user k demands file W_{d_k} . Consider a given file W_n , and for this file we can classify all the subsets of I_K of cardinality $t+1$ according to the number of users requesting file W_n , which we denote as t_n . Only t_n 's in the range

$$\max(1, t+1 - (K - m_n)) \leq t_n \leq \min(t+1, m_n) \quad (14)$$

are relevant to us. The cardinality- $(t+1)$ subsets given above in fact correspond to the subsets used to determine the linear combination in the scheme of [1], and the decomposition indeed starts from the transmissions associated with such subsets. The delivery transmissions related to file W_n in this scheme are as follows. For a given file W_n and a fixed subset $\mathcal{A} \subseteq \bar{I}^{[n]}$ where $|\mathcal{A}| = t+1 - t_n$, the following summations in \mathbb{F}_{2^m} are sent:

$$\bigoplus_{k \in \mathcal{B}} W_{n, \mathcal{A} \cup \mathcal{B} \setminus \{k\}}, \quad |\mathcal{B}| = t_n, \quad \ell^{[n]} \in \mathcal{B} \subseteq I^{[n]}. \quad (15)$$

In Fig. 2 we include an illustration of the relation of various sets; note \mathcal{A} can be the empty set, but cannot have more than t elements. Clearly there are a total of $\binom{m_n-1}{t_n-1}$ such summations for each fixed (n, \mathcal{A}) . In fact, the files segments in 15 for a fixed (n, \mathcal{A}) are all in the following set

$$\mathcal{W}_{n, \mathcal{A}} \triangleq \{W_{n, \mathcal{S}} : \mathcal{S} \cap \bar{I}^{[n]} = \mathcal{A} \text{ and } |\mathcal{S}| = t\}. \quad (16)$$

In contrast to (15), decomposing directly the delivery transmissions of the scheme in [1], particularly the following group,

$$\bigoplus_{k \in \mathcal{A} \cup \mathcal{B}} W_{d_k, \mathcal{A} \cup \mathcal{B} \setminus \{k\}}, \quad |\mathcal{B}| = t_n, \quad \mathcal{B} \subseteq I^{[n]}. \quad (17)$$

leads to the additional transmissions

$$\bigoplus_{k \in \mathcal{B}} W_{n, \mathcal{A} \cup \mathcal{B} \setminus \{k\}}, \quad |\mathcal{B}| = t_n, \quad \ell^{[n]} \notin \mathcal{B} \subseteq I^{[n]}. \quad (18)$$

The transmissions in (18) are however redundant given the information in (15). To illustrate the point of the redundant transmissions in the decomposition, consider again our running example of $(N, K) = (3, 4)$, $t = 2$, and for the demand (A, A, B, C) , for which the transmissions of the code in [1] are as given in (3). Clearly decomposing these transmission directly will yield $A_{3,4}$ twice, one of which is indeed redundant. However, the redundant transmission of $A_{3,4}$ is obtained by decomposing $A_{3,4} + B_{2,4} + C_{2,3}$ which requires $\mathcal{B} = \{2\}$, and since $\ell^{[1]} = 1 \notin \mathcal{B}$, this redundant transmission will be eliminated as it is not included in (15).

The following lemma, which generalizes a similar lemma in [11], can be used to prove that the symbols as given above in (18) are indeed redundant given the transmissions in (15). In fact, this lemma is slightly more general than we need for this purpose, however we state it in this general form since it is useful for more general code constructions.

Lemma 3 (Redundancy Reduction Lemma). *Fix a demand vector (d_1, \dots, d_N) , and denote the set of files being actively demanded as $\mathcal{D} \triangleq \{n \in I_N | d_n > 0\}$. Designate a subset $\mathcal{T} \subseteq \mathcal{D}$ as the variable set, and $\bar{\mathcal{T}} = \mathcal{D} \setminus \mathcal{T}$ as the fixed set, from which we further fix an arbitrary subset $\mathcal{A} \subseteq \cup_{n \in \bar{\mathcal{T}}} I^{[n]}$ where $|\mathcal{A}| \leq t$. Let $\mathcal{U} \triangleq \cup_{n \in \mathcal{T}} \{\ell^{[n]}\}$. Fix also a non-negative integer vector $\{t_n \leq m_n | \sum_{n \in \mathcal{T}} t_n = t+1 - |\mathcal{A}|\}$. Let $\mathcal{Q} \subseteq (\cup_{n \in \mathcal{T}} I^{[n]}) \setminus \mathcal{U}$ be any subset such that $|\mathcal{Q} \cap I^{[n]}| = t_n$ for $n \in \bar{\mathcal{T}}$. Let $\mathcal{V}_{\mathcal{Q}}$ be the family of all subsets \mathcal{V} of $\mathcal{Q} \cup \mathcal{U}$ such that $\mathcal{V} \cap I^{[n]} = t_n$ for $n \in \mathcal{T}$. The following equation holds*

$$\bigoplus_{\mathcal{V} \in \mathcal{V}_{\mathcal{Q}}} \bigoplus_{k \in \mathcal{V}} W_{d_k, \mathcal{V} \cup \mathcal{A} \setminus \{k\}} = 0. \quad (19)$$

Proof. Observe that

$$LHS = \bigoplus_{u \in \mathcal{U}} \bigoplus_{\mathcal{V} \in \mathcal{V}_{\mathcal{Q}}} \bigoplus_{k \in \mathcal{V}: d_k = d_u} W_{d_u, \mathcal{V} \cup \mathcal{A} \setminus \{k\}} \quad (20)$$

Consider any fixed $u \in \mathcal{U}$, and partition $\mathcal{V}_{\mathcal{Q}}$ by indexing through the parts $\hat{\mathcal{V}}_u \triangleq \{\mathcal{V} \cap (\cup_{n \in \mathcal{S} \setminus \{d_u\}} I^{[n]}) : \mathcal{V} \in \mathcal{V}_{\mathcal{Q}}\}$, and $\tilde{\mathcal{V}}_u \triangleq \{\mathcal{V} \cap I^{[d_u]} : \mathcal{V} \in \mathcal{V}_{\mathcal{Q}}\}$ separately. We thus have

$$\bigoplus_{\mathcal{V} \in \mathcal{V}_{\mathcal{Q}}} \bigoplus_{k \in \mathcal{V}: d_k = d_u} W_{d_u, \mathcal{V} \cup \mathcal{A} \setminus \{k\}} = \bigoplus_{\hat{\mathcal{V}} \in \hat{\mathcal{V}}_u} \bigoplus_{\tilde{\mathcal{V}} \in \tilde{\mathcal{V}}_u} \bigoplus_{k \in \tilde{\mathcal{V}}} W_{d_u, \hat{\mathcal{V}} \cup \mathcal{A} \cup (\tilde{\mathcal{V}} \setminus \{k\})} \quad (21)$$

It is obvious that $\hat{\mathcal{V}} \triangleq \tilde{\mathcal{V}} \setminus \{k\}$ is a subset of cardinality $t_n - 1$, and thus there are precisely two subsets $\tilde{\mathcal{V}} \in \tilde{\mathcal{V}}_u$ of cardinality t_n such that $\hat{\mathcal{V}} \subseteq \tilde{\mathcal{V}}$, *i.e.* $\hat{\mathcal{V}} \cup \{k\}$ and $\hat{\mathcal{V}} \cup \{\ell^{[d_u]}\}$, because the set $\mathcal{Q} \cup \{\ell^{[d_u]}\}$ has cardinality $t_n + 1$ and $\tilde{\mathcal{V}}$ is its subset. In other words, for each $(u, \hat{\mathcal{V}}, \mathcal{A}, \hat{\mathcal{V}})$, the term $W_{d_u, \hat{\mathcal{V}} \cup \mathcal{A} \cup \hat{\mathcal{V}}}$ appears exactly twice in this binary extension field summation, and thus they cancel out each other in any binary extension field. \square

Note the subset \mathcal{A} which specifies all other files demands has been fixed in (18), and thus remain invariant in enumerating the summands. To see the transmissions in (18) are redundant given those in (15), let $\mathcal{T} = \{n\}$, and thus $\mathcal{U} = \{\ell^{[n]}\}$. Choose an arbitrary set $\mathcal{Q} \subseteq I^{[n]} \setminus \{\ell^{[n]}\}$ such that $|\mathcal{Q}| = t_n$. Then from (19) we can write

$$\bigoplus_{k \in \mathcal{Q}} W_{n, \mathcal{Q} \cup \mathcal{A} \setminus \{k\}} = \bigoplus_{\mathcal{V} \in \mathcal{V}_{\mathcal{Q}}: \mathcal{Q} \neq \mathcal{V}} \bigoplus_{k \in \mathcal{V}} W_{n, \mathcal{V} \cup \mathcal{A} \setminus \{k\}}, \quad (22)$$

which states that transmissions from decomposing a transmission associated with a set without the leaders can be written as binary summations of those with the leaders. This was exactly our claim.

The next lemma show that there is no further redundancy in the transmissions in (15) to be removed.

Lemma 4. *The linear combinations in (15) are linearly independent.*

Proof. We show that even if we remove some variables in the linear combinations, the remaining linear combinations are still linearly independent. Notice that the linear combinations are formed by a total of $\binom{m_n}{t_n}$ file symbols from $\mathcal{W}_{n, \mathcal{A}}$, and there are a total of $\binom{m_n - 1}{t_n - 1}$ linear combinations in (15). Let us keep only the variables (*i.e.*, file segments) $W_{n, \mathcal{A} \cup \mathcal{B} \setminus \{\ell^{[n]}\}}$ for $\mathcal{B} \subseteq I^{[n]}$ and $|\mathcal{B}| = t_n$. Clearly there are a total of $\binom{m_n - 1}{t_n - 1}$ distinct symbols, one in each of the linear combinations of (15), which implies that they are linearly independent. This completes the proof of the lemma. \square

Another relevant but straightforward observation is that for different (n, \mathcal{A}) pairs, the transmitted groups $\mathcal{W}_{n, \mathcal{A}}$ of linear combinations in (15) are disjoint, because a difference in a component in the pair will result in different W_n , or a difference in \mathcal{A} , respectively. Thus there is no further redundancy to be removed even when we consider the union of different groups, *i.e.*, the transmitted symbols are linearly independent in the delivery phase.

4.3 Proof of Correctness and Performance: All Files Requested

Before providing a more rigorous proof of the correctness and performance of the code, we first provide some intuition in a less formal manner. In this delivery phase, each user essentially will collect a sufficient number of symbols involving those in the cache (the components of the linear

combinations), and then resolve all these symbols to their uncoded form, if we can make the union of the linear combinations in the cache and those collected linearly independent. With cached symbols all resolved, the recovery becomes simple, and a delivery strategy similar to that in [1] will suffice. Indeed, in the delivery strategy, we are simply transmitting decomposed symbols of those in [1] with redundancy removed, and thus recovery is trivially guaranteed after the cached file segments have been recovered to their uncoded form.

Since the caching strategy is the same as that in [10], the performance of this alternative scheme hinges only on the total number of transmitted symbols, which can be seen easily that for each n to be

$$\sum_{t_n=\max(1,t+1-(K-m_n))}^{\min(t+1,m_n)} \binom{K-m_n}{t+1-t_n} \binom{m_n-1}{t_n-1} = \binom{K-1}{t}, \quad (23)$$

where the equality can be checked to be true by viewing the LHS as the number of ways to choose t balls from a total of $(K-1)$ balls, but from two separate groups of cardinalities $K-m_n$ and m_n-1 , respectively. The total amount of transmissions is thus $N \binom{K-1}{t}$, which after normalizing by the size of the file $\binom{K}{t}$ gives exactly the transmission rate R in (4).

It remains to show that the scheme is indeed correct, *i.e.*, each user can collect sufficient symbols in the delivery phase to completely recover the individual uncoded symbols in the cache, and moreover, they are linearly independent together with the cached contents. For this purpose, without loss of generality, we only consider user k_0 which requests W_{n_0} . Let us consider the number of symbols this user collects that are formed by W_n , $n = 1, \dots, n_0-1, n_0+1, \dots, N$. In the decomposition, for an arbitrary subset of I_K of cardinality $t+1$ which k_0 is a member, the decomposed part formed using segments of file W_n will be collected. To be more precise, recall that for a fixed (n, \mathcal{A}) pair, where $n \neq n_0$, $\max(1, t+1-(K-m_n)) \leq t_n \leq \min(t+1, m_n)$, and $k_0 \in \mathcal{A} \subseteq \cup_{i \neq n} I^{[i]}$ where $|\mathcal{A}| = t+1-t_n$, all the symbols in (15) can be collected from the transmissions. These symbols are linearly independent, and also linearly independent across different (n, \mathcal{A}) pairs. Summing over all valid (n, \mathcal{A}) pairs gives

$$\sum_{n=2}^N \sum_{t_n=\max(1,t+1-(K-m_n))}^{\min(t+1,m_n)} \binom{K-m_n-1}{t-t_n} \binom{m_n-1}{t_n-1} = (N-1) \binom{K-2}{t-1},$$

symbols to collect, where the equality is again by the same ball counting argument. Moreover, these symbols are all linear combinations of the symbols

$$\{W_{n,\mathcal{S}}, \text{ for all } n \in I_N \setminus \{n_0\}, \text{ and all } \mathcal{S} \text{ such that } k_0 \in \mathcal{S} \text{ and } |\mathcal{S}| = t\},$$

or in other words, they are formed using the same set of symbols as those in user k_0 's cache. Together with the $P_0 - P$ symbols in user k_0 's cached, we have a total of P linear combinations of a total of P symbols.

It only remains to show that we can choose the coefficients for the cached content such that these P linear combinations are indeed linearly independent. There are two possible approaches to accomplish this. We provide here an explicit construction based on linearized polynomial, and omit the alternative proof which only shows the existence of a valid coefficient assignment; interested readers can refer to [10] for more details where both approaches were used.

Consider encoding the P file segment symbols in (12) with a (P_0, P) systematic rank metric code, as described in Section 2.3, and use the $(P_0 - P)$ parity symbols as the cached contents. By Lemma 2, as long as the matrix G which encodes the rank metric code symbols (including both the

parity parts which are cached, and the systematic parts which are used to form linear combinations during delivery) into the P linear combinations from both the cache memory and the collected symbols at user k_0 is full rank (rank P), the proof is complete. This is in fact trivially true, since this matrix is block diagonal, with each block being full rank: the first $(P_0 - P)$ -by- $(P_0 - P)$ matrix is an identity matrix, and the others are simply the binary coding matrix for each group of symbols derived from (15), each of which has already been shown to be full rank in Lemma 4.

4.4 Proof of Correctness and Performance: Some Files Requested

A substitution-based technique was given in [10] to address the case when only a strict subset of the files are being requested by the users, and the corresponding code requires only minor modification from the case when all files are requested. We shall extend this technique in the setting of the alternative code and show that the same delivery rate can be achieved.

Let us consider the case when $N^* < N$ files are requested. Without loss of generality, let us assume that the first N^* files are being requested, and $I^{[n]}$, m_n and $\bar{I}^{[n]}$ are defined similarly as in the last subsection, but only for $n = 1, 2, \dots, N^*$. To describe the transmission strategy, we first find another set of “enhanced demands”, parametrized by $\dot{I}^{[1]}, \dot{I}^{[2]}, \dots, \dot{I}^{[N]}$, where all files are being requested; *i.e.*, $|\dot{I}^{[n]}| \geq 1$ for $n = 1, 2, \dots, N$. Additionally, these enhanced demands must satisfy the following properties:

- $|\dot{I}^{[n]}| = 1$ for $n = N^* + 1, \dots, N$;
- For any $k \in \{1, 2, \dots, K\}$, if $k \in I^{[n]}$, then either (1) $k \in \dot{I}^{[n]}$, or (2) $k \in \dot{I}^{[n']}$ for some $n' \in \{N^* + 1, \dots, N\}$ and $k \neq \ell^{[n]}$; for case (2), denote the mapping from n' to n as $f(n') = n$, and denote the mapping from n' to k as $u(n')$.

We also write $|\dot{I}^{[n]}| = \dot{m}_n$ for simplicity. The enhancement replaces some users’ requests with requests for other files that originally are not being requested, and each of these files is now being requested by only one user in the enhanced version. Such an enhancement can always be found under the condition $N \leq K$.

The transmission strategy is as follows:

1. For each file W_n , $n = 1, 2, \dots, N^*$, transmit the linear combinations as given in (15) following the **enhanced demands**;
2. For each n , $n = N^* + 1, \dots, N$, perform the following operations. For each $\mathcal{S} \subseteq I_N$, where $u(n) \notin \mathcal{S}$ and $|\mathcal{S}| = t$,
 - (a) Transmit $W_{f(n), \mathcal{S}}$, if $\ell^{[f(n)]} \in \mathcal{S}$ and it has not been transmitted in its uncoded form;
 - (b) Otherwise, transmit $W_{n, \mathcal{S}}$.

Because in the enhanced demands, there is only one user who requests any W_n , $n = N^* + 1, \dots, N$, thus each transmission of the scheme in [1] has no more than one component from W_n . It follows that the transmissions $\{W_{n, \mathcal{S}} | u(n) \notin \mathcal{S} \text{ and } |\mathcal{S}| = t\}$ are indeed the decomposed parts from the transmitted symbols in [1] for the enhanced demands. Some of these transmissions are simply replaced by $W_{f(n), \mathcal{S}}$ in the strategy given above.

We next show that the delivery strategy is correct and the performance remains the same as in the previous case. The latter statement is in fact trivially true, by observing that the number of delivery transmissions is the same as that for the enhanced demands, which reduces it to the previous case when all files are requested. To show the code is indeed correct, we show that with

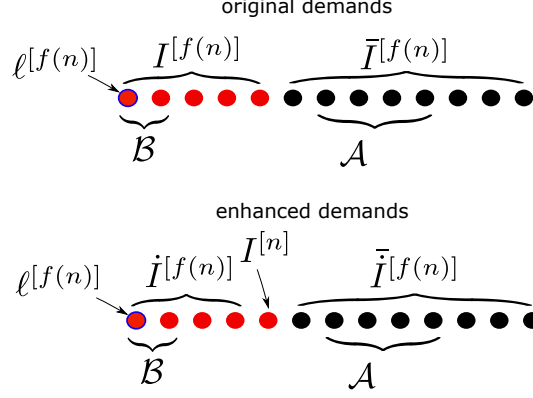


Figure 3: The relation of various sets in the original and enhanced demands.

the transmissions, all the users can resolve the symbols in the cache to their uncoded forms, and then show that each user can also recover any required segments that are not in the cache (as components of the linear combinations). For this purpose, without loss of generality consider user k_0 who requests file W_{n_0} in the original demands. Two cases need to be treated separately: the first case is when in the enhanced demands, user k_0 also requests W_{n_0} , and the second case is when in the enhanced demands, user k_0 in fact requests another file $W_{n_0^*}$, *i.e.*, $f(n_0^*) = n_0$ and $u(n_0^*) = k_0$.

For the first case, we observe that for $n = N^* + 1, \dots, N$ there are indeed $(N - N^*) \binom{K-2}{t-1}$ symbols to collect, in the forms of $W_{f(n), \mathcal{S}}$ or $W_{n, \mathcal{S}}$, for $\{\mathcal{S} | n_0 \in \mathcal{S} \text{ and } u(n) \notin \mathcal{S} \text{ and } |\mathcal{S}| = t\}$. As long as these symbols, together with the linear combinations in the cache of user k_0 , are linearly independent, user k_0 can resolve all the symbols its cache to the uncoded format. This follows from the observation that at the end of the delivery transmissions, whenever a segment $W_{f(n), \mathcal{S}}$ is transmitted in step 2, the whole transmission group $\mathcal{W}_{f(n), \mathcal{S} \cap \bar{I}[f(n)]}$ will be recoverable from the delivery transmissions alone. This fact can basically be shown using the line of arguments in the proof of Lemma 4. More precisely, observe that in step 1 the following $\binom{\bar{m}_{f(n)}-1}{t-|\mathcal{S} \cap \bar{I}[f(n)]|}$ symbols were transmitted in this group

$$\bigoplus_{k \in \mathcal{B}} W_{f(n), (\mathcal{S} \cap \bar{I}[f(n)]) \cup \mathcal{B} \setminus \{k\}}, \quad |\mathcal{B}| = t + 1 - |\mathcal{S} \cap \bar{I}[f(n)]|, \quad \ell[f(n)] \in \mathcal{B} \subseteq I[f(n)]. \quad (24)$$

The uncoded transmissions in 2(a) have the following $\binom{\bar{m}_n-1}{t-|\mathcal{S} \cap \bar{I}[f(n)]|-1}$ symbols from the same group

$$W_{f(n), (\mathcal{S} \cap \bar{I}[f(n)]) \cup \mathcal{B} \setminus \{k\}}, \quad k \neq \ell[f(n)], \quad |\mathcal{B}| = t + 1 - |\mathcal{S} \cap \bar{I}[f(n)]|, \quad \ell[f(n)] \in \mathcal{B} \subseteq I[f(n)]. \quad (25)$$

There are a total of $\binom{\bar{m}_n}{t-|\mathcal{S} \cap \bar{I}[f(n)]|}$ symbols in the transmission group $\mathcal{W}_{f(n), \mathcal{S} \cap \bar{I}[f(n)]}$, and the transmissions in (24) and (25) are sufficient to resolve them, because after eliminating the symbols in (25) from the linear combinations of (24), each summation in (24) has only one unknown as

$$W_{f(n), (\mathcal{S} \cap \bar{I}[f(n)]) \cup \mathcal{B} \setminus \{\ell[f(n)]\}}, \quad |\mathcal{B}| = t + 1 - |\mathcal{S} \cap \bar{I}[f(n)]|, \quad \ell[f(n)] \in \mathcal{B} \subseteq I[f(n)]. \quad (26)$$

Additionally, the transmissions of any uncoded $W_{n, \mathcal{S}}$ for $n = N^* + 1, \dots, N$ are indeed linearly independent from everything else. Thus user k_0 can recover the cached contents to its uncoded form. At this point user k_0 can use the transmissions in step 1 for the enhanced demands to recover all the missing segments of file n_0 .

For the second case, we observe for W_n , $n = 1, 2, \dots, N^*$, user k_0 can collect a total $N^* \binom{K-2}{t-1}$ useful symbols in step 1 of the transmission. For files W_n , $n = N^*+1, \dots, n_0^*-1, n_0^*+1, \dots, N$, through the same line of argument as the first case, it can be shown straightforwardly $(N - N^* - 1) \binom{K-2}{t-1}$ useful symbols can be collected, moreover they are all linearly independent, and linearly independent with the cached content, thus the user k_0 can recover the cached symbols to their uncoded form. At this point is also clear user k_0 can recover all file segments of W_{n_0} , since all the missing segments can in fact be completely decoded from transmissions in step 1 and step 2(a) as we just showed.

4.5 Reduction in Alphabet Sizes

In the scheme given in [10] based on rank metric code and MDS code, a sufficient alphabet size was given as \mathbb{F}_{q^m} where $q \geq 2^{\binom{K-N+1}{\max(\lfloor (K-N+1)/2 \rfloor, t)}}$ and $m \geq P_o$. In the alternative version of the code we propose here, which is also based on rank metric code, but with the MDS code component replaced by a binary code, we can use an alphabet size of 2^m where $m \geq P_o$. This is a significant reduction in the alphabet size for this explicit code construction.

In [10], it was also shown if one is willing to replace the rank metric code using a generic linear code, then there exists a code in a much smaller alphabet, and in fact an alphabet size $S(K, N)N!$ suffices, where

$$S(K, N) = \frac{1}{N!} \sum_{j=1}^N (-1)^{N-j} \binom{N}{j} j^K. \quad (27)$$

The code proposed in the current work can be understood as one such case. We could also follow the same line of proof of existence using the new code given in this work, but this does not reduce the alphabet size further from $S(K, N)N!$.

5 Partial Decompositions for New Codes

It is now clear that the scheme in [10] can be obtained by firstly decomposing the delivery transmissions in [1] and removing redundancy, and accordingly decreasing amount of cached contents from fully uncoded (*i.e.*, linear combinations whose rank is the same as the base dimension) to fully coded. This naturally leads to a view of partial decomposing the delivery transmissions, where the strategy in [10] and that in [11] are simply the two extremes. We next present one particular example to show that this is indeed effective and can in fact lead to a new tradeoff point, and then briefly discuss the general scheme.

5.1 A New Tradeoff Point for $(N, K) = (3, 4)$

Consider again the case $(N, K) = (3, 4)$, where each file has 6 segments in the alphabet \mathbb{F}_{2^m} where $m \geq 17$. Each user caches 8 linear combinations of the information symbols of the same uncoded file segments, as in both [1] and [10]. This can be obtained using a $(17, 9)$ systematic rank metric code, and caching the parity symbols. We claim that delivering a total of 5 coded symbols is sufficient in the second phase, which gives an achievable pair $(M, R) = (4/3, 5/6)$. This would be strictly better than $(4/3, 23/27)$ achieved by the lower convex hull of the schemes [10, 11, 13], which gives the best known achievability result in the literature; see Fig. 4 for an illustration. For completeness, we also include a computer-generated outer bound which was obtained in a separate work [19]. Interestingly, both $(M, R) = (3/8, 2)$ given by the code in [13] and $(M, R) = (4/3, 5/6)$ obtained in this work are in fact on this outer bound, and thus optimal.

We next prove our claim. Due to the symmetry in the code, we only need to consider the cases when the demands are (A, A, B, C) , (A, A, B, B) , (A, A, A, C) and (A, A, A, A) .

- For the demand (A, A, B, C) . Instead of fully decomposing the transmissions in (3), we now partially decompose them as follows

$$\begin{aligned} &A_{2,3} + A_{1,3} + B_{1,2}, A_{2,4} + A_{1,4} + C_{1,2}, \\ &B_{1,4} + C_{1,3}, B_{2,4} + C_{2,3}, A_{3,4}. \end{aligned} \quad (28)$$

User 1 collects $B_{1,4} + C_{1,3}$, and thus together with the 8 cached linear combinations, can resolve all the symbols in (5); now user 1 essentially has uncoded cache contents, and thus can recover the missing file segments of A (which are $A_{2,3}, A_{2,4}, A_{3,4}$) using the remaining transmissions. User 2 can use a similar strategy. User 3 can collect $A_{3,4}$ and together with the 8 cached linear combinations, can resolve all the symbols in the cached linear combinations to the uncoded form. Now user 3 essentially has uncoded cache contents, and can use the rest of transmissions to recover the missing segments of file B . A similar argument holds for user 4.

- For the demand (A, A, B, B) , we can transmit the following

$$\begin{aligned} &A_{2,3} + A_{1,3} + B_{1,2}, A_{2,4} + A_{1,4} + B_{1,2}, \\ &B_{1,4} + B_{1,3}, B_{2,4} + B_{2,3}, A_{3,4}. \end{aligned} \quad (29)$$

User 1 can collect $B_{1,4} + B_{1,3}$ in order to resolve the cached symbols, and the decoding is similar to the previous case. It is also obvious user 3 and user 4 can indeed recover file B .

- For the demand (A, A, A, C) , we can transmit

$$\begin{aligned} &A_{2,3} + A_{1,3} + A_{1,2}, A_{2,4} + A_{1,4} + C_{1,2}, \\ &A_{1,4} + C_{1,3}, A_{2,4} + C_{2,3}, A_{3,4}. \end{aligned} \quad (30)$$

The decoding strategies of other users are similar to the previous cases, and let us only consider user 3 as an illustration. It can use $A_{3,4}$ to resolve the symbols in the cache, then recover $A_{1,4}$ from $A_{1,4} + C_{1,3}$, $A_{2,4}$ from $A_{2,4} + C_{2,3}$, and $A_{1,2}$ from $A_{2,3} + A_{1,3} + A_{1,2}$.

- For the demand (A, A, A, A) , we can transmit

$$\begin{aligned} &A_{2,3} + A_{1,3} + A_{1,2}, A_{2,4} + A_{1,4} + A_{1,2}, \\ &A_{1,4} + A_{1,3}, A_{2,4} + A_{2,3}, A_{3,4}. \end{aligned} \quad (31)$$

Using a similar strategy as above, it is seen that all users can indeed recover file A .

It is clear that we have used the substitution technique discussed in the previous sub-section, and the core of the delivery strategy is for the demands (A, A, B, C) , and all other cases simply follow from substituting the appropriate file symbols into the corresponding enhanced demands.

5.2 A General Decomposition Approach

We can easily generalize the example above, by partially decomposing the transmissions of the delivery code [1], and correspondingly reducing the amount of cached contents which are properly generated using rank metric code. More precisely, in this general scheme

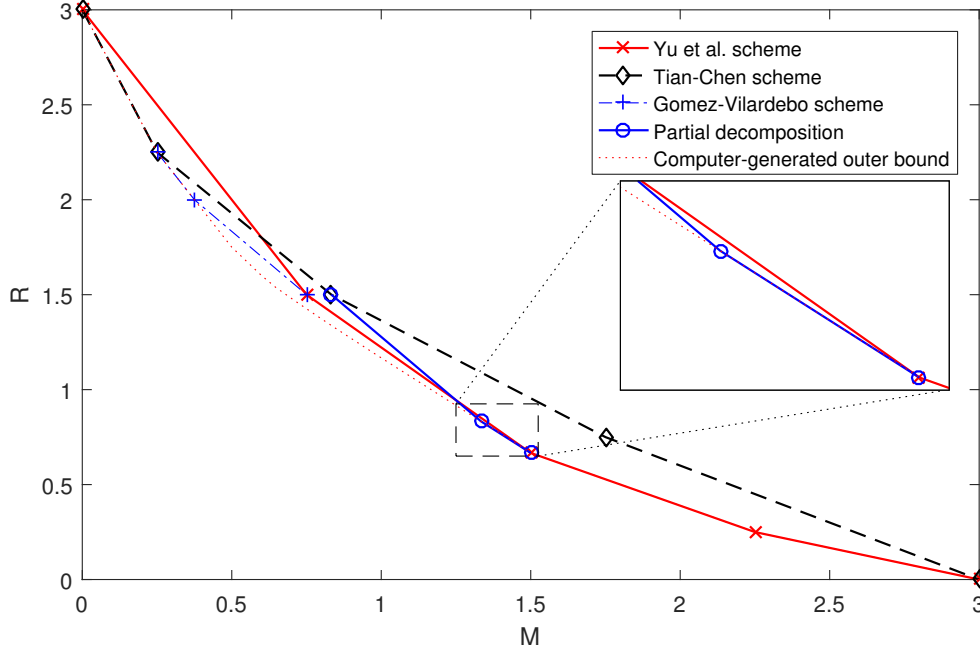


Figure 4: A new tradeoff point for $(N, K) = (3, 4)$.

- During the prefetching, each user collects the symbols in (12), and then caches P^* linear combinations of them using the parity symbols of a $(P^* + P, P)$ rank metric code;
- For a demand vector (d_1, d_2, \dots, d_K) , the original transmissions of the scheme [1] are decomposed in a pattern of choice, and the redundancy can be removed using Lemma 4.
- For the transmission associated with a $(t + 1)$ -subset in the scheme [1], if a user is in this subset, then the corresponding symbols are collected by this user.
- Each user first resolves the cached symbols to the uncoded form, and then uses the rest of the delivery transmissions to recover the missing segments.

This strategy in principle allows us to find more new tradeoff pairs, by adjusting P^* , and choosing good decomposition patterns. However this generic description of the strategy does not lead to a closed form formula for a decomposed strategy, and optimizing within all possible decompositions is also difficult. There are many demand types, and within each demand type, there are many ways one can decompose the transmissions. Although for a fixed decomposition pattern, we can rely on Lemma 4 to remove the redundancy, and the use of rank metric code in prefetching essentially reduces the requirement on linear independence to a simple counting, this task is still quite daunting. The number of different demand types and decomposition possibilities increase quickly as (N, K) become large. The verification of correct decoding for a given parameter (N, K) and a particular set of decompositions can be done using a computer program, but at this point we are not able to find a method to analytically determine the optimal decompositions.

6 Conclusion

We propose a connection between the caching strategy in [10] and that in [11], by decomposing the delivery transmissions in [11] to yield those in [10]. This first allows us to provide an alternative

coding strategy to that in [10], where the delivery part uses only binary codes, and secondly, it allows us to view the caching strategy in [10] and that in [11] as the two extremes of a more general scheme. We provided details for one intermediate code in this general scheme, which gives a new tradeoff unknown in the literature for the case $(N, K) = (3, 4)$. Characterizing the performance of this general scheme appears difficult, which is part of our ongoing work.

References

- [1] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” *IEEE Trans. on Information Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.
- [2] —, “Decentralized coded caching attains order-optimal memory-rate tradeoff,” *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug. 2015.
- [3] —, “Coded caching with nonuniform demands,” in *Proc. of INFOCOM Workshops 2014*, Toronto, Canada, Apr.-May 2014, pp. 221–226.
- [4] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, “Online coded caching,” in *Proc. of 2014 IEEE International Conference on Communications (ICC)*, Sydney, Australia, Jun. 2014, pp. 1878–1883.
- [5] N. Karamchandani, U. Niesen, M. Maddah-Ali, and S. Diggavi, “Hierarchical coded caching,” in *Proc. of 2014 IEEE International Symposium on Information Theory (ISIT)*, Honolulu HI, Jul. 2014, pp. 2142–2146.
- [6] Z. Chen, P. Fan, and K. B. Letaief, “Fundamental limits of caching: Improved bounds for small buffer users,” *arXiv:1407.1935*, 2014.
- [7] S. Sahraei and M. Gastpar, “ k users caching two files: An improved achievable rate,” *arXiv:1512.06682*, 2015.
- [8] M. Amiri, Q. Yang, and D. Gunduz, “Coded caching for a large number of users,” *arXiv:1605.01993*, 2016.
- [9] K. Wan, D. Tuninetti, and P. Piantanida, “On caching with more users than files,” *arXiv:1601.06383*.
- [10] C. Tian and J. Chen, “Caching and delivery via interference elimination,” *IEEE Transactions on Information Theory*, submitted, see *arXiv:1604.08600*, and *Proceedings of IEEE ISIT 2016*.
- [11] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “The exact rate-memory tradeoff for caching with uncoded prefetching,” *arXiv:1609.07817*, 2016.
- [12] M. M. Amiri and D. Gunduz, “Fundamental limits of caching: Improved delivery rate-cache capacity trade-off,” *IEEE Trans. on Communications*, vol. 65, no. 2, pp. 806–815, Feb. 2017.
- [13] J. Gómez-Vilardebó, “Fundamental limits of caching: Improved bounds with coded prefetching,” *arXiv preprint arXiv:1612.09071*, 2016.
- [14] E. M. Gabidulin, “Theory of codes with maximum rank distance,” *Probl. Peredachi Inf.*, vol. 21, no. 1, pp. 3–16, 1985.

- [15] R. Lidl and H. Niederreiter, *Finite fields (Encyclopedia of mathematics and its applications)*. Cambridge University Press, 1997.
- [16] R. Koetter and F. Kschischang, “Coding for errors and erasures in random network coding,” *IEEE Trans. on Information Theory*, vol. 54, no. 8, pp. 3579–3591, Aug. 2008.
- [17] N. Silberstein, A. Rawat, and S. Vishwanath, “Error-correcting regenerating and locally repairable codes via rank-metric codes,” *IEEE Trans. on Information Theory*, vol. 61, no. 11, pp. 5765–5778, Nov. 2015.
- [18] C. Tian, B. Sasidharan, V. Aggarwal, P. V. Kumar, and V. Vaishampayan, “Layered exact-repair regenerating codes via embedded erasure correction and block designs,” *IEEE Trans. on Information Theory*, vol. 61, no. 4, pp. 1933–1947, Apr. 2015.
- [19] C. Tian, “Symmetry, outer bounds, and code constructions: A computer-aided investigation on the fundamental limits of caching,” *IEEE Transactions on Information Theory*, submitted, see also *arXiv:1611.00024*.